Microsoft Malware Prediction

Team : The Brainiacs

Vaibhavi Tikone
IIIT Bangalore
vaibhavi.tikone@iiitb.org

Hritik Arora
IIIT Bangalore
hritik.arora@iiitb.org

Souvik Das
IIIT Bangalore
souvik.das@iiitb.org

ABSTRACT

Nowadays, the Internet is widely available, just one second of global online activity is packed full of events, from communication with others to data storage to entertainment options. And with it, the danger of malicious attacks by cybercriminals have increased. These attacks are done via Malware (short for 'Malicious – Software.") and have resulted in billions of dollars of financial damage. This makes the prevention of malicious attacks an essential part of the battle against cybercrime.

In this paper, we are applying machine learning algorithms to predict the malware infection rates of computers based on its features. We are using supervised machine learning algorithms and gradient boosting algorithms. We have collected a publicly available dataset, which was divided into two parts, one being the training set, and the other will be the testing set. After conducting four different experiments using the algorithms, it has been discovered that LightGBM is the best model with an AUC (Area Under the ROC Curve) Score of 0.71848.

Keywords- malware prediction, machine learning algorithm, lgbm, k-fold, decision tree, microsoft malware prediction dataset

1 PROBLEM STATEMENT/BUSINESS PROBLEM DESCRIPTION

The goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. The malware industry continues to be a well-organized, well-funded market dedicated to evading traditional security measures. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways. With more than one billion enterprise and customers, Microsoft takes this problem very seriously and is deeply invested in improving security. As one part of their overall strategy for doing so, Microsoft is challenging the data science community to develop techniques to predict if a machine will soon be hit with malware. As with their previous, Malware Challenge (2015), Microsoft is providing Kagglers with an unprecedented malware dataset to encourage open-source progress on effective techniques for predicting malware occurrences.

2 DATASET DESCRIPTION

The dataset that has been used for our project is the Microsoft Malware Prediction Dataset that has been used in the Microsoft Malware Prediction Competition posted on Kaggle this year.

The size of the dataset is massive, with the training dataset named 'train.csv' containing 567730 rows, and the testing dataset named 'test.csv' containing 243313 rows. There are 82 features contained in the dataset, with most being categorical, of which 23 are numerically encoded to protect the privacy of the information contained in the dataset. Each row in this dataset corresponds to a machine, uniquely identified by a 'MachineIdentifier'. 'HasDetections' is the ground truth and indicates that Malware was detected on the machine. Using the information and labels in train.csv, you must predict the value for 'HasDetections' for each machine in test.csv.

3 DATA PROCESSING TASKS

1. VISUALIZATIONS AND INFERENCES

   (a) Distribution of data over the class labels in the dataset:

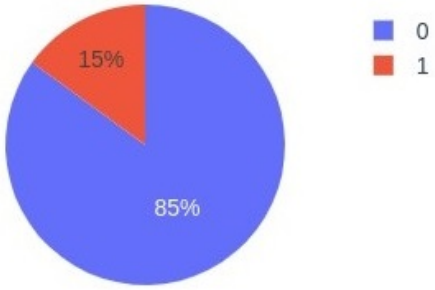

Figure 1: Target class distribution

This shows that our dataset is highly imbalanced. Only 15% of the training dataset containing 'HasDetection' value 1. So, a smaller number of machines have been detected as malware infected.

   (b) Correlation Between Categories:

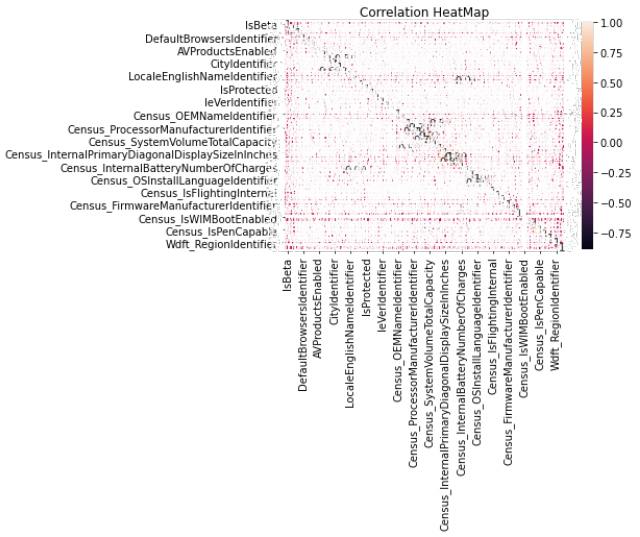

Figure 2: Correlation matrix

A correlation heat map is based on data analysis which uses colors by showing a bar graph uses width and height for a data visualization tool. From the correlation matrix, we can infer that our features are independent. Also, our dataset is highly imbalanced so we must handle it accordingly.

2. DATA CLEANING AND PREPROCESSING

Our data has total 82 features, so let's explore the data-

(a) Missing Values:

	Features	Unique_values	missing_values_percent	type
28	PuaMode	1	99.983971	object
41	Census_ProcessorClass	3	99.624293	object
8	DefaultBrowsersIdentifier	599	94.821306	float64
68	Census_IsFlightingInternal	2	82.798513	float64
52	Census_InternalBatteryType	28	70.515386	object
71	Census_ThresholdOptIn	2	63.139521	float64
75	Census_IsWIMBootEnabled	1	63.043172	float64
31	SmartScreen	14	36.592218	object
15	OrganizationIdentifier	43	30.822222	float64
29	SMode	2	6.752682	float64

Figure 3: Missing values

We can see the features 'PuaMode' and 'Census_ProcessorClass' are almost empty. In our dataset there are 44 features which contains null values.

(b) Skews:

	Features	Unique_values	skewness	type
5	IsBeta	2	99.999295	int64
27	AutoSampleOptIn	2	99.997534	int64
28	PuaMode	1	99.983971	object
65	Census_IsPortableOperatingSystem	2	99.946101	int64
35	Census_DeviceFamily	3	99.810649	object
41	Census_ProcessorClass	3	99.624293	object
33	UacLuaenable	5	99.294735	float64

Figure 4: Skews

We can see the features 'IsBeta', 'AutoSampleOptIn', 'PuaMode' and some others are highly skewed. In our dataset there are 24 features which are having skewness of more than 90 percent. There are many techniques to handle the skewness, but there is well said quote that handling skewness means you are playing with information you have.

(c) Imputations:

In imputation techniques, we fill null values with some normalized value of that feature. In our dataset some features are numeric datatype, and some are object datatype. But after having close look to the dataset, most of the features are categorical (non-ordinal) regardless of its datatype. So, we tried different techniques, like replacing with mode of that feature, mean of that feature, using Imputer library itself. But for replacing by mode value of that feature gave us best score.

(d) Encoding techniques:

The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model.

Since most machine learning models only accept numerical variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model can understand and extract valuable information. We tried different types of encoding techniques like, one hot encoding, label encoding, frequency encoding and target encoding; among all frequency encoding gave best score for our model.

3. FEATURE ENGINEERING

(a) Dimensionality Reduction:

We used feature selection method for dimensionality reduction task. We removed highly skewed and highly null valued features such as 'PuaMode', 'IsBeta'. We listed 17 such features, and reduced the size from 82 to 65. But we realized that, keeping all the features, infact helps the model to learn well, and when we tested the prediction for test data, we came to know that all the features helped the model to converge well.

(b) New Features:

In our dataset there are so many features and as all are related, we derived new features. There is 'Census_PrimaryDiskTotalCapacity' and 'Census_SystemVolumeTotalCapacity', from which we derived new feature 'memory_remain'. Same way there are some features which gives information about display resolution, from those also we derived 'pixel' feature. But the model which gave best score worked well without these additional features. Microsoft already have derived so many new features and added to the dataset.

4 MODEL TRAINING

Our objective is to predict class of 'hasDetection' 0-safe, 1-malware detected.So, we need classification algorithms to train the model. For best score we must tune the parameters of the model. When we talk of tuning models, we specifically mean tuning hyperparameters.

There are two types of parameters in machine learning algorithms. The key distinction is that model parameters can be learnt directly from the training data while hyperparameters cannot. And that is why most of the data scientists spend half of the time on tuning the hyperparameters for the model. We have mostly worked with the LightGBM algorithm, which is new, but has many useful applications in Malware Prediction.

LightGBM is a gradient boosting framework that uses based tree learning algorithms. Compared to other algorithms where trees grow horizontally, meaning it grows level-wise, LightGBM trees grow vertically, which means that it grows leaf-wise. We took learning rate 0.01. We chose values of parameters like n_estimators and num_leaves accordingly, which gave us best score. After so many trials, we got best score for n_estimators equal to 6000 with early stopping rounds equal to 100, and num_leaves 250. We did not set max_depth parameter value as it was giving lower score for same model.

The final AUC score is 0.71848 for the same model.

5 Feature Importance

We plotted the graph of feature importance for our best model. Here is the figure of 20 key features of our best model.

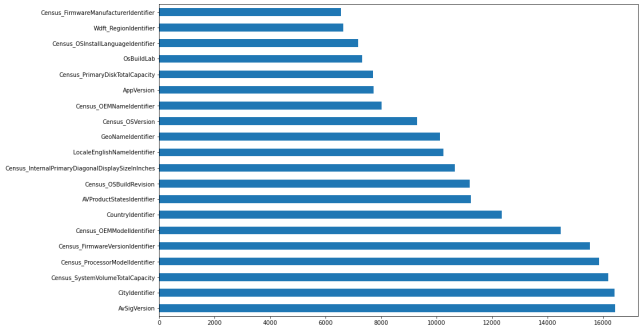

Figure 5: Feature Importance plot

6 Results

1. EVALUATION METRICS:

   Our experimentation requires the use of a training dataset, where the information from it is used to train the model. The trained model is then used to predict the value of 'HasDetections' in the test dataset.

   Since the data is imbalanced F1 score and ROC will be used as the parameters for judging the performances of the models. The AUC score is an indicator of the performance of models and is the perfect metric to base our observations on. AUC Scores below 0.5 are bad, and scores between 0.5 and 0.7 are average. Scores of 0.7 and above are good scores. We are using it as the defining metric for this experiment, as it is the metric used by Microsoft Corporation in its data competitions pertaining to Malware Prediction.

   KERNEL DENSITY ESTIMATE(KDE) PLOT is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

   Here is the KDE plot of train and test dataset for predicted probability from trained model-

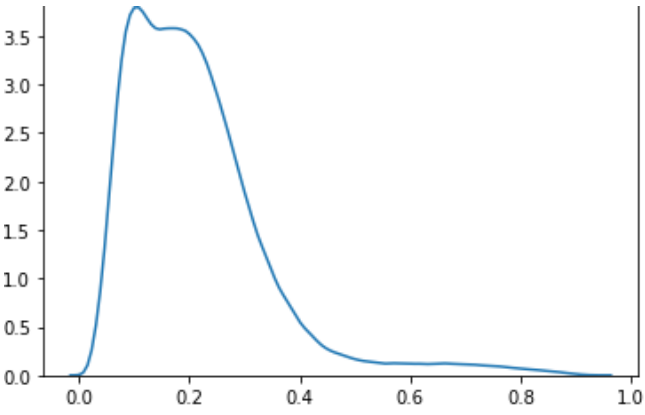

Figure 6: Training data

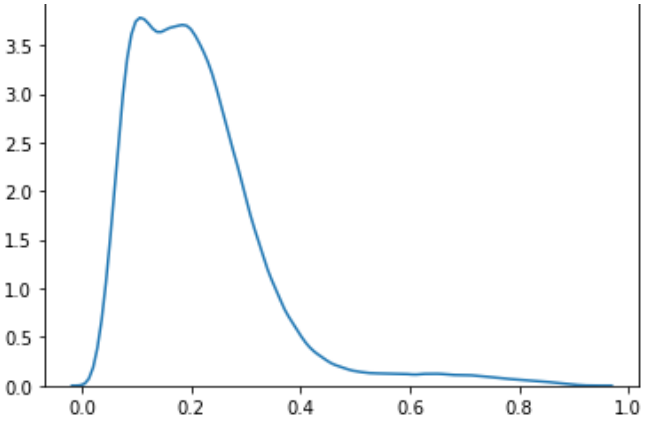

Figure 7: Testing data

2. COMPARISON OF DIFFERENT MACHINE LEARNING TECHNIQUES:

   We started with the basic classifier- logistic regression. We used Sklearn libraries for that. We tuned parameters and checked the accuracy. Logistic regression is linear classifier, and when we removed those 17 features from our dataset, and trained the model, this gave us 0.65 AUC score. This is better than expected. But our intuition said we can do better than this. As we have a greater number of features, we then jumped to complex models like LightGBM, Random forest, and XGBoost.

   We first tried random forest, but it did not perform well, it gave 0.63 AUC score. We also tried XGBoost model and it gave 0.68 AUC score after tuning parameters. It took too much time for us to tune the parameters of XGBoost, so we jumped to alternative algorithm of XGBoost, which is LightGBM.

   We trained the LightGBM model with the dataset having 65 features. This model initially gave 0.63 AUC curve, but after tuning the parameters we were able to score 0.69. This score was better than all prior models. We analyzed the dataset again to boost the score, and we removed only 5 features which were highly skewed, highly null valued or contained only 1 unique value. After changing this approach, we trained the LightGBM model with same parameters; this model gave us 0.71848 AUC score.

7 Conclusion

In our research, we conducted three experiments with the use of three different algorithms - LightGBM, Random forest, and logistic regression on the dataset we collected from the Microsoft Malware Prediction Dataset.

We used Area Under the ROC Curve (AUC) Score as the defining metric for our experimentation. After the analysis of the results of all the experiments, LightGBM fitted into a Sparse Matrix provided the best results, as it obtained the highest AUC Score, and had great efficiency, taking a comparatively low amount of time to run. This shows the viability of LightGBM as the leading Gradient Boosting algorithm in the prediction of malware infection rates in machines. It has immense potential to be implemented in future Malware Prediction and Protection systems.

8 PROJECT LINK

The link for our project files are available at:
https://drive.google.com/drive/folders/1kHuZ5asmrGvX9XMT_yaTtx_dbSFIErB?usp=sharing

REFERENCES

[1] Zawad, Safir,Evan "Analysis of Malware Prediction Based on Infection Rate Using Machine Learning Techniques", 2020.

[2] N. DuPaul,"Malware",Veracode,[Online]. Available: https://www.veracode.com/security/malware

[3] Guolin Ke, Qi Meng "LightGBM: A Highly Efficient Gradient Boosting Decision Tree",2017

[4] Gavrilut D., Cimpoesu M., Anton D., and Ciortuz L., "Malware Prediction Using Machine Learning", International Multiconference on Computer Science and Information Technology, 2009.

[5] Baset, M, "Machine Learning for Malware Detection", 2016.

[6] LightGBM documentation available: https://lightgbm.readthedocs.io/en/latest/index.html

[7] Nikhil Sai, Kaggle Competition "Microsoft Malware Prediction" available: https://www.kaggle.com/c/Malware-detection-NS